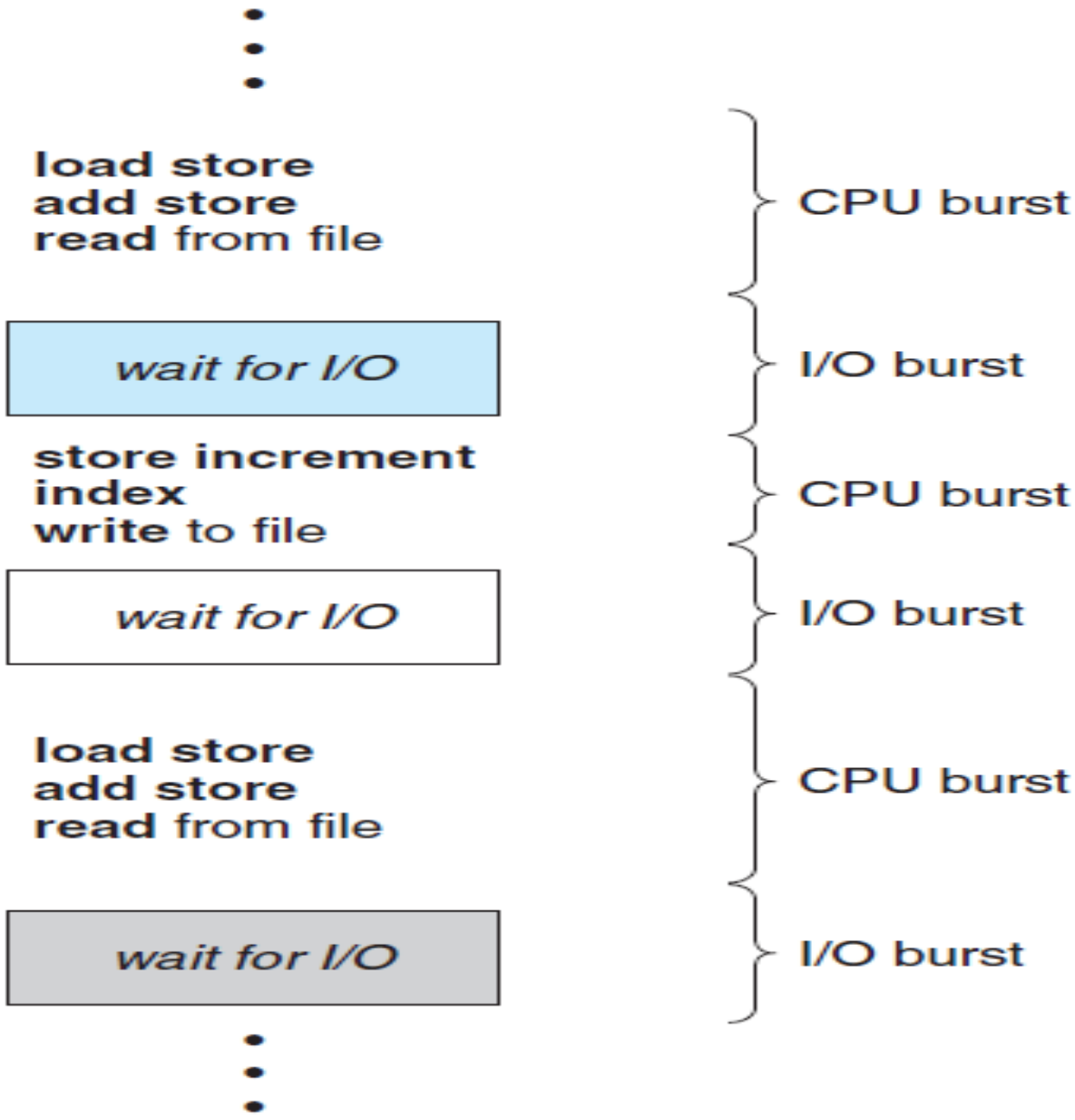
الباب الثالث

جدولة العمليات

المجدول : Short-term scheduler

الغاية من **تعدد البرامج هو ابقاء المعالج في حالة شغل دائم** وذلك بتناوب المعالج على تنفيذ عدة عمليات. **ولتحقيق ذلك لابد من تمييز وفهم الاتي:**

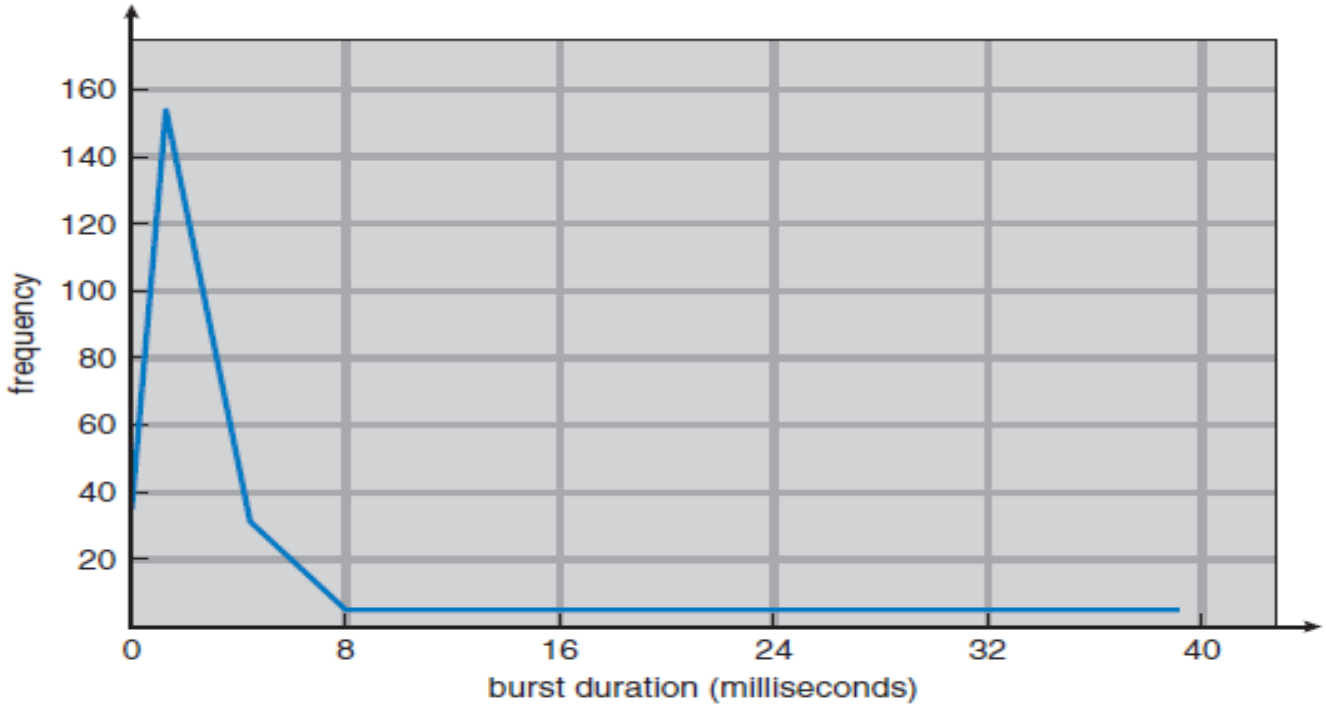
- **سلسلة البيرست CPU - I/O Burst** : يتطلب تنفيذ اي **عملية تنفيذ سلسلة من اوامر الادخال والاخراج وكذلك اومر حسابية** لا تتعلق باستخدام الموارد الخارجية (الشكل 4.1).


load store
add store
read from file
wait for I/O
store increment
index
write to file
wait for I/O
load store
add store
read from file
wait for I/O
CPU burst
I/O burst
CPU burst
I/O burst
CPU burst
I/O burst

الشكل 4.1

تنفيذ العملية (Process) يتألف من تكرار تنفيذ **التعليمات الحسابية (CPU-Bound)** يليها تنفيذ التعليمات الخاصة **بتعليمات الادخال والاخراج (I/O-Bound)**. أي ان العملية تتقلب ما بين هاتين الحالتين الى ان يتم ايقافها.

قياس تكرار زمن دوام هاتين الحالتين وجد انه يختلف باختلاف **نوع الحاسوب**، وايضا يختلف **باختلاف العملية**، ولكنه في الاجمال يتخذ توزيعا قريبا من الشكل 4.2.
حيث ان هذا التوزيع يتميز بعدد كبير من التعليمات الحسابية القصيرة، وعدد صغير من تعليمات I/O الطويلة.



الشكل 4.2

Figure 6.2 Histogram of CPU-burst durations.

خصائص الجدولة

من المهام الاساسية لنظام التشغيل هو **تخصيص او توزيع الموارد بين العمليات المتعددة**، ومن بين هذه الموارد **توزيع وقت المعالج بين هذه العمليات**.

الية هذا التوزيع تسمى بالجدولة ويراعى في هذه الالية الخصائص الاساسية التالية :

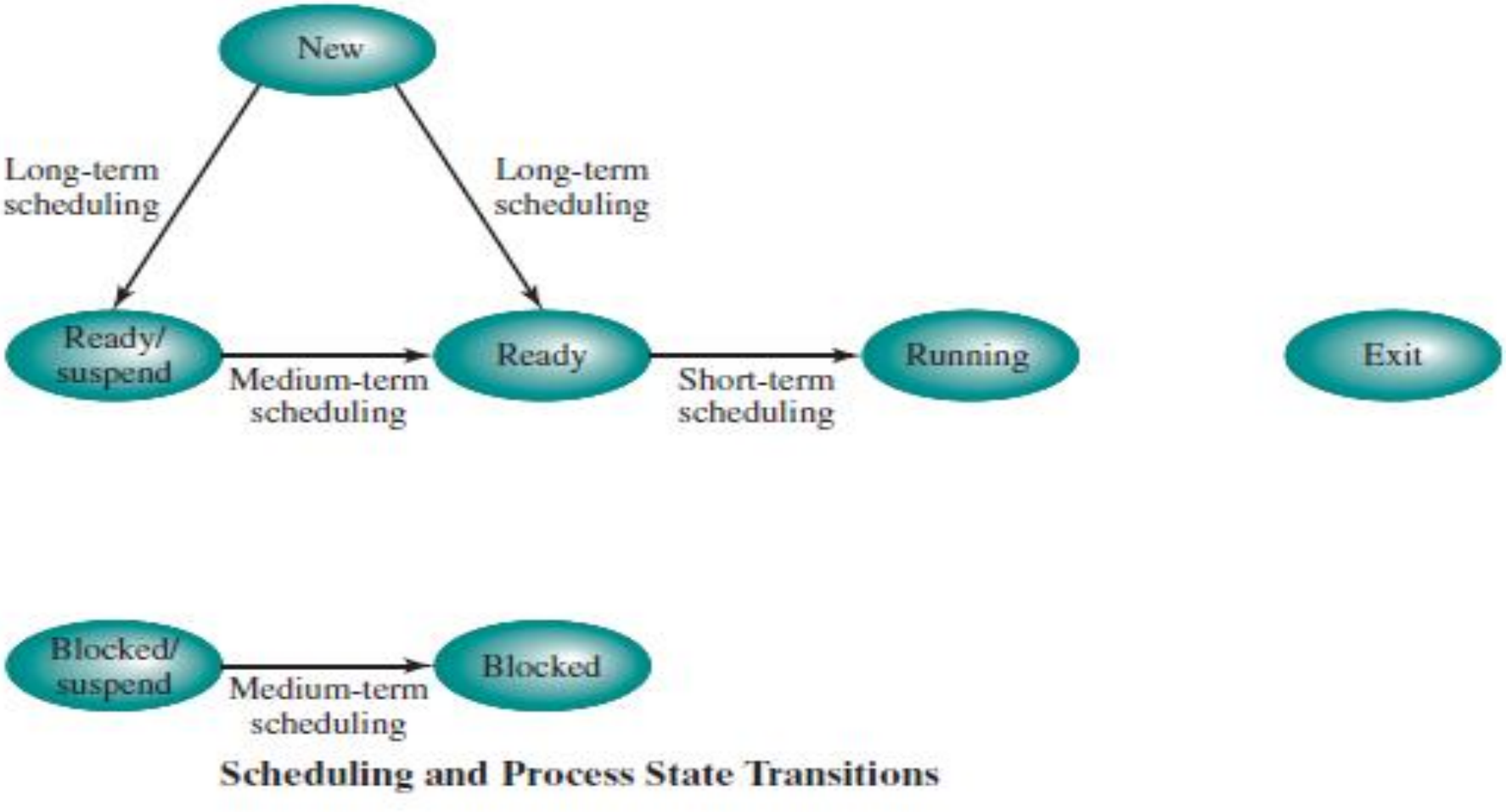
- **العدل (Fairness) :** توزيع **وقت المعالج** بطريقة تؤمن نفس الفرص لكل العمليات.
- **قلة الحرمان (Lack of starvation)** : ضمان **عدم احتكار المعالج من قبل بعض العمليات**. هذا الاحتكار يؤدي الى حرمان بعض العمليات الاخرى، بمعنى بقاءها زمن اطول تنتظر فرصة التنفيذ.
- **الاستخدام الامثل لزمن المعالج (Efficient use of processors' time) :** استخدام زمن المعالج بطريقة فعالة.
- **التقليل من الزمن الضائع او الاضافي (Low overheads) :** الزمن الذي **يقضيه المعالج في الانتقال من عملية الى اخري** والذي يترتب عليه حفظ حالة العملية الحالية واستجلاب حالة العملية التالية.

انواع الجدولة

بصفة عامة يوجد **اربعة انواع من الجدولة** – **ثلاثة** منها تعتبر من **جدولة المعالج** و**واحدة** تهتم بجدولة **اجهزة الادخال والاخراج**. **هذه الانواع مبينة في الجدول الاتي:**

الجدولة طويلة المدى (Long-term scheduling)	وتتم عند اضافة عملية الى مخزن (Pool) **العمليات المراد تنفيذها** وهي موجودة على القرص الصلب في حالة نظام الدفعات
الجدولة متوسطة المدى (Medium-term scheduling)	وهي تهتم بإضافة **العمليات الموجودة جزئيا او كاملة بالذاكرة** اي انها تتعلق بعملية التبديل (Swapping).
الجدولة قصيرة المدى (Short-term scheduling)	وهي تهتم بتحديد العمليات (الموجودة بالذاكرة) التي سوف يتم تنفيذها – أي **العمليات الموجودة بقائمة الجاهزية (READY STATE)**
جدولة I/O (I/O scheduling)	وهي تهتم بجدولة **العمليات التي تنتظر اجهزة الادخال والاخراج.**

يبين **الشكل 4.3** الحالات التي تمر بها العمليات وعلاقتها بالأنواع المختلفة للجدولة ومتي يمكن لجدولة ما ان تؤدي وظيفتها.



الشكل 4.3

جدولة المعالج – المجدول (CPU Scheduler)

وهي عملية انتقال المعالج من عملية الى اخرى, لكي يبقى في حالة شغل في اغلب الاوقات, حيث يتم اختيار العملية عن طريق ما يعرف **بالجدولة قصيرة المدى (Short term Scheduler)** وذلك باختيار احدى العمليات **الموجودة بالذاكرة**.

يمكن تصنيف الجدولة بانها **قابلة للتوقف (الاخلاء قسريا)** او **غير قابلة للتوقف** يطلق عليها على التوالي **(Preemptive Scheduler)** و **(nonpreemptive scheduler)**.

- ففي الجدولة التي تتميز بأنها **غير قابلة للتوقف nonpreemptive** تترك العملية في **حالة التنفيذ** الى ان تدخل في حالة **الانتظار او الجمود Blocked –** (وذلك بطلب **عملية I/O** او تنتظر عملية اخرى لكي تكمل عملها) او ان تنهي تنفيذها ذاتيا و تلقائيا نتيجة لأنها اكملت مهمتها.
- بينما في حالة الجدولة التي تتميز بأنها **قابلة للإخلاء القسري (Preemptive Scheduling)**، فان العملية **تَترك المعالج بمجرد انتهاء الزمن المخصص** وفقا للشريحة الزمنية المعطاة.

متى نحتاج للجدولة ؟

نحتاج الى الجدولة وفقا للظروف الاتية:

1. عند الانتقال من حالة **التنفيذ** الى حالة **الانتظار او الجمود - BLOCKED** (طلب I/O, انهاء العملية الابن).
2. عند الانتقال من حالة **التنفيذ** الى حالة **الجاهزية – READY** (مقاطعة انتهاء شريحة الزمن Time-out).
3. عند الانتقال من حالة **الانتظار او الجمود BLOCKED** - الى حالة **الجاهزية** (انتهاء طلب I/O).
4. عند **انهاء العملية.**

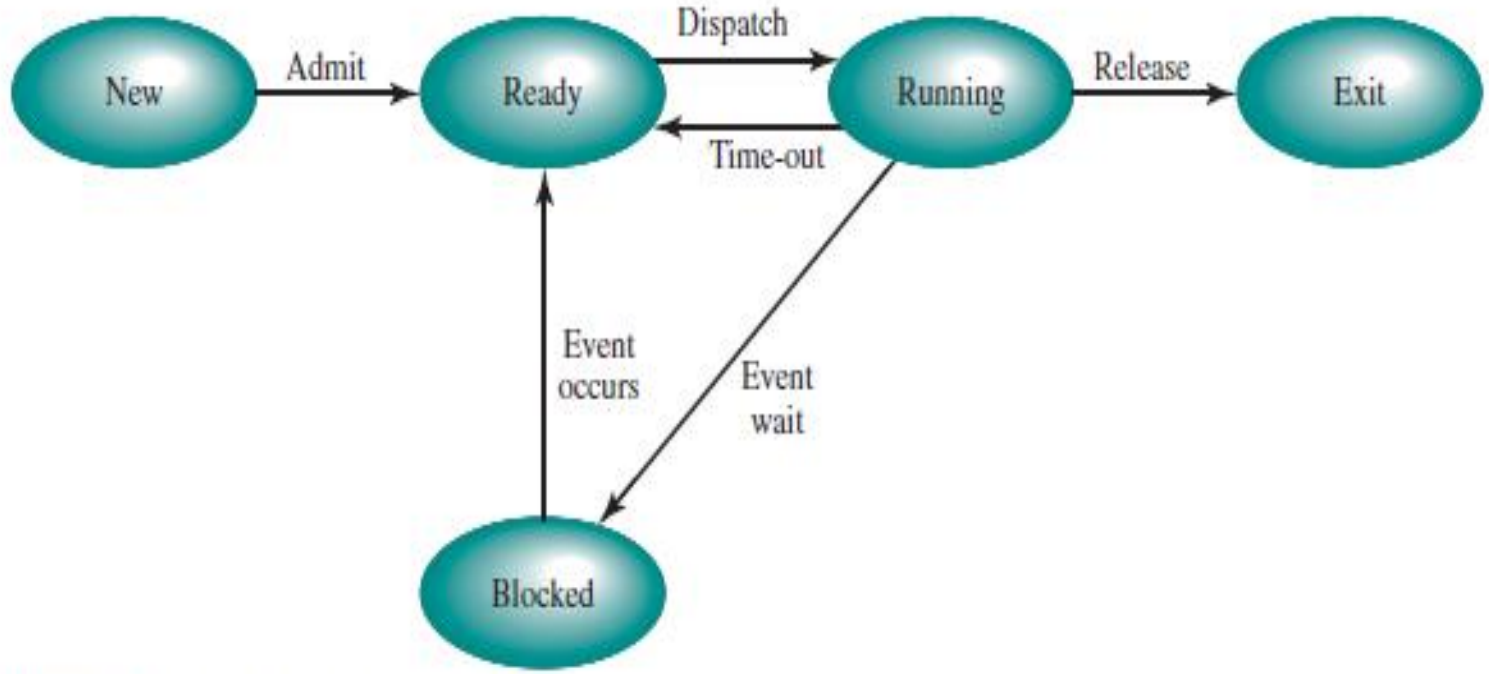
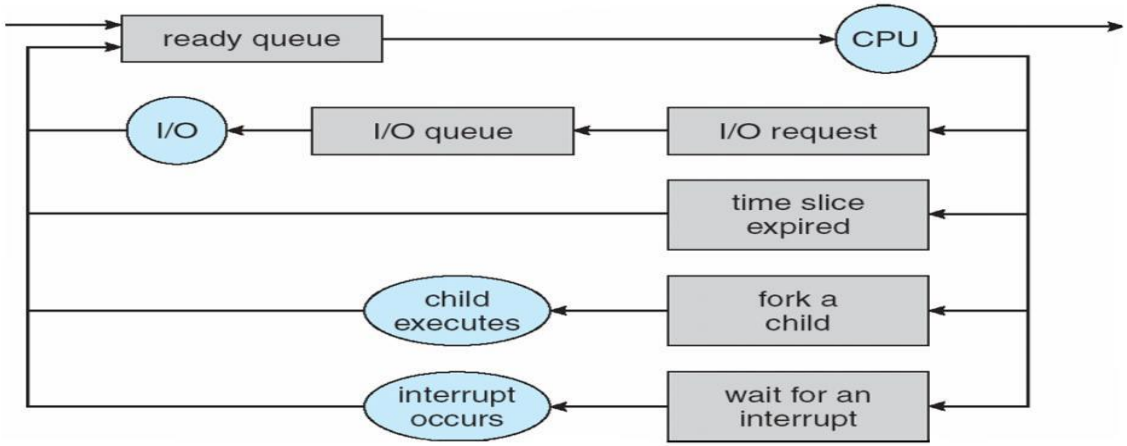
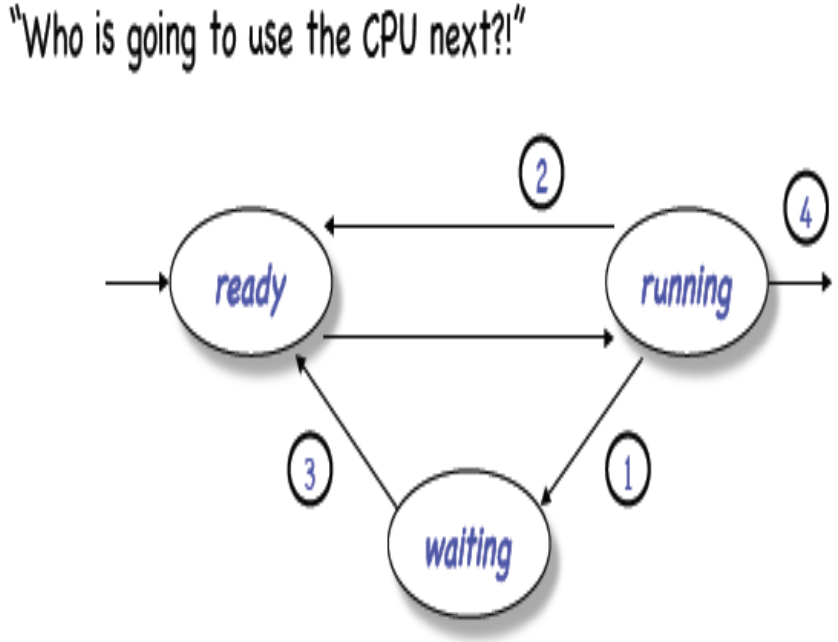


Figure 3.6 Five-State Process Model

الشكل الاتي يوضح انسياب العمليات من وضعية الى اخرى





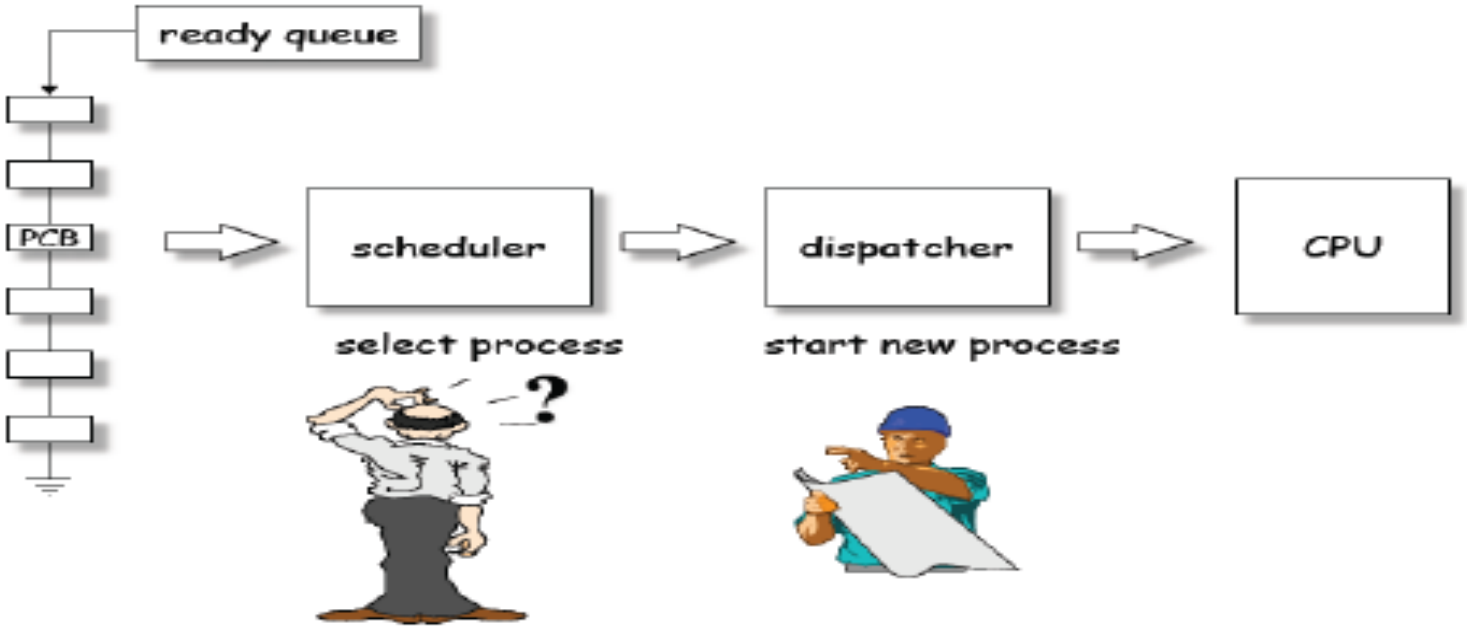
- في الحالات **1 و 4** ستخرج العملية **إجباريا** إما **لانتهائها** أو لأنها **تحتاج دخل أو خرج**، في هذه الحالة لابد للمجدول من اختيار عملية بديلة، فليس لديه خيار آخر. في هذه الحال نقول أن المجدول **non-preemptive** .
- أما **3،2** فإن العملية تخرج **اختيارا**، وقد تواصل، هنا للمجدول حق الاختيار في إخراجها أو لا. هنا نقول أن المجدول . **preemptive**

المرسل Dispatcher

هو **الوحدة المسئولة على اعطى عملية ما لوحدة المعالج وفقا للجدولة قصيرة المدى** وتقوم بالمهام الاتية:

- جلب **بيانات سياق الانتقال** (Switching Context).
- تغيير صيغة النظام من **صيغة النظام** (Kernel mode) الى **صيغة المستخدم** (user mode).
- الانتقال الى **النقطة المخصصة في العملية** لاستئناف العمل.

يجب ان يكون المرسل (Dispatcher) سريع لأنه **يتم استدعائه اثر كل عملية انتقال** ويعرف هذا الوقت بالـ **Dispatch latency** . وفي بعض الانظمة قد **يصبح المرسل هو نظام الجدولة الرئيسي** حيث يقوم بالإضافة لمهامه بمهام الجدولة.



قواعد الجدولة (Scheduling Criteria)

طرق الجدولة لها عدة خصائص يمكن عن طريقها **المفاضلة** بينها وهي عبارة عن **قياسات بالاداء** حسب مفهوم المستخدم (User-oriented) او النظام (System-oriented) وتشمل **الخصائص الاتية**:

- **زمن استخدام المعالج System-oriented (CPU Utilization)** : وهي **نسبة تشغيل المعالج** حيث من المفروض ان يعمل المعالج بنسبة 100% في الحالة المثالية ولكن في الواقع فان النسبة تتروح بين 40%-90%.
- **كمية الشغل او الانتاجية System-oriented (Throughput):** وهي عبارة عن **عدد العمليات** التي يتم تنفيذها خلال زمن معين.
- **الزمن المستغرق او الكلي User-oriented (Turnaround Time)** : الزمن المستغرق من بداية تقديم العملية الى نهاية تنفيذها وهو يتضمن **مجموع** زمن **الانتظار للدخول للذاكرة** (Load time), زمن الانتظار في **قائمة الجاهزية**, وزمن **التنفيذ في المعالج** وزمن **تنفيذ الادخال والاخراج (I/O)** أي زمن الانتظار في حالة الانتظار او الجمود (Blocked).
- **زمن الانتظار (Wait Time)** : وهو الزمن المستغرق **في بقاء العملية في قائمة الجاهزية** ويتم جمع الفترات الزمنية في حالة تكرار وجود العملية في قائمة الجاهزية.
- **زمن الاستجابة User-oriented (Response Time)** : وهو الزمن ما بين **تقديم الطلب وظهور اول النتائج** – وهو يتأثر او يعتمد على سرعة اجهزة الادخال والاخراج. ويستخدم في الانظمة التفاعلية (Interactive Systems).

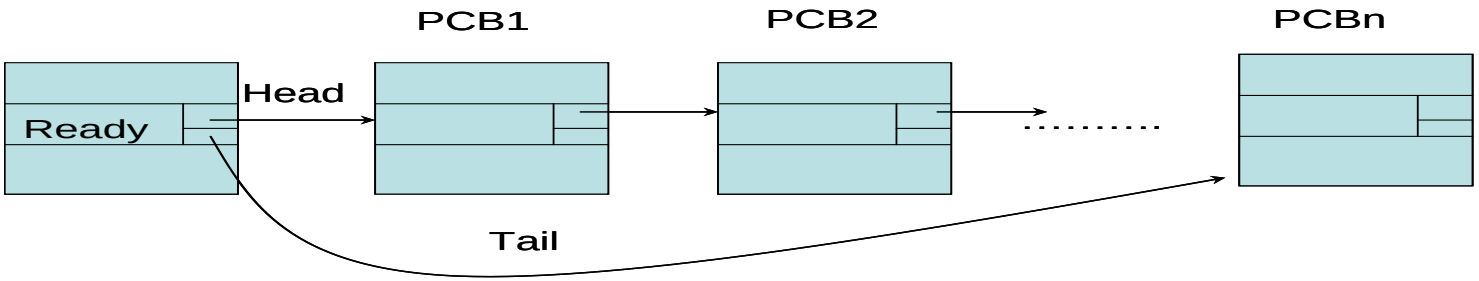
يسعى نظام التشغيل الى **زيادة زمن استخدام المعالج** (CPU Utilization) و زمن **كمية الشغل** (Throughput) **وتقليص الزمن المستغرق** (Turnaround) و **زمن الانتظار** (Waiting time) و**زمن الاستجابة** (Response Time).

في الغالب يتم تحسين متوسط الزمن مع وجود بعض الحالات التي يتم فيها تحسين القيمة الصغرى او العظمى – كما في حالة الانظمة التفاعلية عندما نريد تحسين اداء النظام وتقديم زمن استجابة افضل وذلك بتقليص القيمة العظمى لزمن الاستجابة.

خوارزميات الجدولة (Scheduling Algorithms)

خوارزمية الرتل او الطابور (First-Come, First-Served FCFS)

في هذه الخوارزمية يتم تخصيص المعالج لأول عملية موجودة بالطابور اي بمعنى حسب وصولها لقائمة الجاهزية (Ready state) – الاول فالأول. تصنف هذه الجدولة بالـ **Nonpreemptive scheduling**.
حيث يتم تخزين **تركيبة العملية (PCB)** في قائمة الجاهزية على هيئة **سلسلة** (Linked List) بحيث تخزن العملية القادمة في ذيل السلسلة (Tail) والعملية الاولى على رأس القائمة (Head) كما هو موضح بالشكل الاتي:



متوسط زمن الانتظار عادة طويل.

مثال : المعلومات الاتية تمثل مجموعة من العمليات وصلوا بالترتيب التالي ويتطلبوا زمن تنفيذ (CPU-Burst) موضح بالجدول الاتي

Process	Burst Time
P1	24ms
P2	3ms
P3	3ms

اذا تم وصولهم حسب التسلسل الاتي **P3 ,P2 ,P1** فان

- **متوسط زمن الانتظار** = (27+24+0)/3=**ms17**
- **متوسط الزمن الكلي** (Turnaround time) = (24+27+30)/3 =**ms27**،

واذا كان الوصول حسب الترتيب **P1 ,P3 ,P2** فان

- **متوسط زمن الانتظار**= (6+3+0)/3 = **3 ms**،
- **ومتوسط الزمن الكلي** = (3+6+30)/3= **ms13**

* نلاحظ ان **زمن الانتظار** يتغير بحسب ترتيب وصول العمليات وخاصة عندما يكون زمن التنفيذ متباعد بين مختلف العمليات.

مميزات وعيوب جدولة الطابور

يعتبر اداء هذه الطريقة افضل بالنسبة للعمليات **الطويلة** منه للعمليات **القصيرة** لان **العمليات القصيرة سوف تنتظر فترة زمنية طويلة للحصول على المعالج، مما يؤدي الى حالة الحرمان (Suffer Starvation).**

تعتبر هذه الطريقة ايضا **لصالح العمليات التي تتميز بكثرة التعليمات الحسابية (CPU-Bound)**، لأنها لن تخرج الا اذا انتهت من عملها، الامر الذي يؤدي الى انتظار العمليات التي تتميز بكثرة تعليمات الادخال/الاخراج (I/O -Bound)

عادة ما يتم دمج نوعا من الاسبقية (Priority) لتحسين ادائها.

جدولة الشغل الاقصر اولا (Shortest-Job-First (SJF) Scheduling)

تنقسم الخوارزمية إلى نوعين هما :

- **غير قابلة للتوقف Non-preemptive** إذا بدأت عملية ما التنفيذ داخل المعالج لن تتوقف اي عدم انتقال المعالج لأي عملية اخرى الا اذا انتهت العملية الحالية، بحيث أن **العملية التي تحتاج إلى زمن اقل يتم تنفيذها أولا.** هذه الجدولة تعطي زمن انتظار افضل مقارنة بجدولة الرتل او الطابور، أيضاً هذه الجدولة تختار الشغل الاقصر المراد تقديم الخدمة له بشكل يضمن أن هذا الشغل سوف ينتهي و يترك النظام في اسرع وقت ممكن، مما يؤدي إلى تخفيض عدد العمليات المنتظرة و بالتالي تخفيض زمن الانتظار العام.

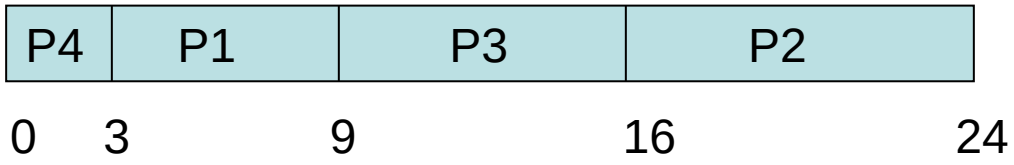
- **قابلة للتوقف Preemptive** إذا وصلت عملية جديدة إلى صف الانتظار وكان زمنها أقصر من الزمن المتبقي للعملية التي يتم تنفذها حاليا بالمعالج سيقوم المجدول بإخراج الأولى وإدخال التي وصلت حديثا.

العيب الأساسي لهذه الجدولة هي **حاجتها لمعلومات على الاقل تقديريه عن إجمالي زمن التنفيذ**، اي الزمن الذي يمكن أن تستغرقه عملية ما ليتم تنفيذها حتى النهاية، وهي معلومات ليست سهلة الحصول عليها وهي قد تكون تقديرية مثلما يحدث في نظام الدفعات (Batch system).

مثال (**غير قابلة للتوقف**) : مجموعة من العمليات بزمن سلسلة المعالج (CPU-Burst) مقدر بالملي ثانية كالاتي:

Process	Burst Time
P1	6ms
P2	8ms
P3	7ms
P4	3ms

باستخدام **جدولة الشغل الاقصر اولا** (**SJF**) فان العمليات تخزن حسب الشكل الاتي:

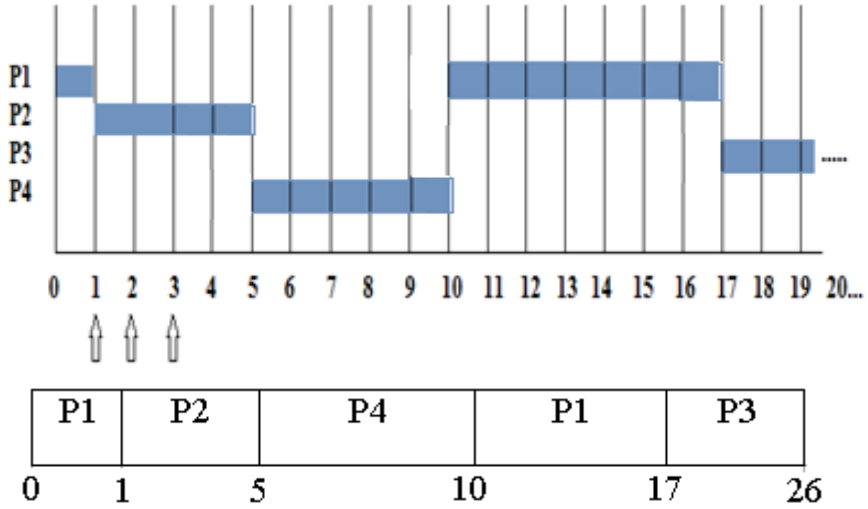


وعليه يكون :

- **متوسط زمن الانتظار** = (0+3+9+16) / 4 = **7ms.**
- **ومتوسط الزمن الكلي** = (0+3+9+16+24) / 4 = **13ms**

مثال (قابلة للتوقف) : اذا كان وصول العمليات الاتية في زمن مختلف ويحتاجوا الى زمن تنفيذ كما هو موضح بالجدول الاتي:

العملية	زمن الوصول	الزمن
P1	0	8
P2	1	4
P3	2	9
P4	3	5



الحل

سيكون ترتيب العمليات داخل المعالج كما يلي:

زمن انتظار العملية = زمن الانتظار الكلي – زمن الوصول وعليه فان :

زمن انتظار العملية P1 = 10- 1 = 9

زمن انتظار العملية P2 = 1 - 1 = 0

زمن انتظار العملية P3 = 17- 2 = 15

زمن انتظار العملية P4 = 5 - 3 = 2

- **ملاحظة: عند تساوي الزمن للعمليات تستخدم خوارزمية الرتل (FCFS).**

متوسط زمن الانتظار = (9+0+15+2)/4= **6.5 م/ث**

متوسط الزمن المستغرق (الكلي) =[(0-17)+(1-5)+(2-26)+(3-10)]/4

= [17+4+24+7]/4=52/4=**13م/ث**

مميزات وعيوب طريقة SJF

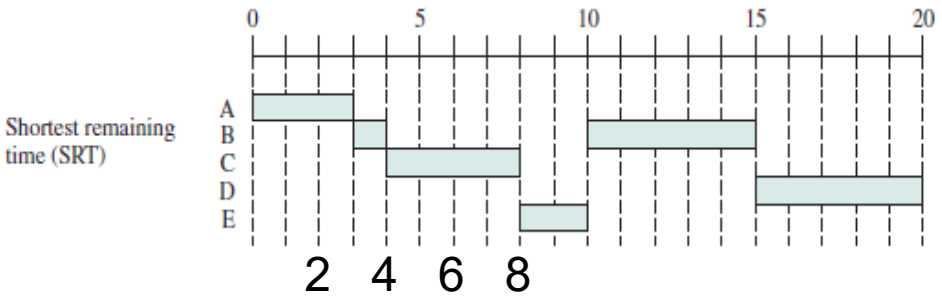
- **العمليات الطويلة تنتظر مدة اطول وقد تتعرض لحالة الحرمان (Starvation)** عندما يتم ادخال عمليات قصيرة باستمرار. بما أن هذه الطريقة تفترض ترتيب الطابور حسب الزمن الاقصر فان العمليات القصيرة تكون دائما في مقدمة الطابور وهذا ما يفسر انتظار العمليات التي تحتاج الى وقت اطول الى فترات طويلة من الزمن للحصول على المعالج.
- **تعتبر هذه الطريقة نوعا من انواع الاسبقية.**
- **اذا كان الزمن التقديري غير صحيح فان نظام التشغيل قد يوقف هذه العملية.**
- هذه الطريقة تهدف الى تشغيل العمليات التالية بأسرع وقت ممكن، **الامر الذي يؤدي الى تقليل عدد العمليات بالنظام، والذي يؤدي بدوره الى تحسين زمن الانتظار اجمالا.**

جدولة الزمن المتبقي الاقصر (Shortest Remaining Time – SRT)

وهي **نسخة الاخلاء القسري من جدولة الشغل الاقصر اولا** وذلك بالاعتماد على تسجيل الزمن المنقضي بعد كل عملية تنفيذ، وهي ايضا تفترض **معرفة زمن التنفيذ عند انشاء العملية**.

حيث يقارن الزمن المتبقي للعملية **X** بالزمن المتبقي للعملية **Y** **قيد التنفيذ** – فاذا كان الزمن المتبقي للعملية **X** اقل من الزمن المتبقي للعملية **Y** **قيد التنفيذ** فانه يتم استبدال العملية **قيد التنفيذ** بالعملية **X** أي الاقل زمنا متبقيا، لان هذه الجدولة ذات اسلوب **الاخلاء القسري**.

مثال – اذا تم وصول العمليات **A, B, C, D, E** في زمن مختلف ويحتاجوا الى زمن تنفيذ كما موضح بالجدول التالي :



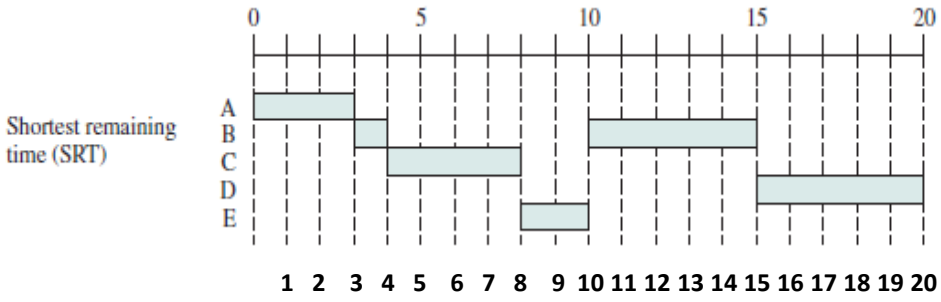
العمليات	زمن الوصول	زمن التنفيذ
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

- هنا العملية **A** يتم تنفذها الى النهاية لان زمن وصول العمليات الاخرى كان متأخرا (انظر الجدول السابق).
- وصلت العملية **B** عند **الثانية 2** الا انها تنتظر لحين اتمام العملية **A** لأنه عند وصول العملية **B** تكون العملية **A** هي الاقصر زمنا متبقيا.
- في **الثانية الرابعة** تصل العملية **C** وهي الاقصر زمنا في هذه اللحظة تعطى الفرصة للتنفيذ لأنها الاقصر زمنا متبقيا ويبقى الزمن المتبقي من العملية **B** **5 ثوان**.
- تستمر العملية **C** لغاية **الثانية 6** حيث تصل العملية **D**، وحيث ان الزمن المتبقي من العملية **C** هو **2 ثانية** اقل من زمن العملية **D**، لذلك تستمر العملية **C** في التنفيذ.
- عند **الثانية 8** تكون العملية **C** قد اتمت مهمتها في اللحظة التي وصلت فيها العملية **E**. وحيث ان العملية **E** هي الاقل زمنا فبالتالي يتم تنفيذها وينتقل المعالج للعملية **B**، ثم العملية **D**.

يمكن حساب متوسط زمن الانتظار حسب الجدول التالي :

العمليات	زمن الانتظار
A	0
B	6+3-2=7
C	4-4=0
D	15-6=9
E	8-8=0
متوسط زمن الانتظار	16/5=3.2ms

اجمالي زمن الانتظار ناقص زمن الوصول



جدولة الاسبقية (Priority Scheduling)

تعتبر جدولة **الشغل الاقصر اولا** (**SJF**) نوع خاص من **جدولة الاسبقية** حيث يمكن اعطاء العملية الاسبقية بنسبة (**1/t**) حيث **t** تمثل زمن التنفيذ لعملية ما.

في جدولة الاسبقية يتم **اعطاء كل عملية اسبقية** من **0-N**.

يمكن اعتبار **الترتيب التصاعدي** يمثل اسبقية العمليات من **الاعلى الى الادنى**.

وبالمثل فانه يمكن اعتبار **الترتيب التنازلي** يمثل اسبقية العمليات من **الادنى الي الاعلى**.
بمعنى انها تعتمد على نوع النظام المستخدم.

في هذا الباب فان الاسبقية تعطى على اساس **الترتيب التصاعدى** حيث يمثل **الرقم 0** العملية ذات **الاسبقية الاعلى** **والرقم 7** العملية ذات **الاسبقية الادنى**.

يمكن ان تكون جدولة الاسبقية **قابلة للتوقف او غير قابلة للتوقف**.

مثال : لدينا 5 عمليات وصلت عند **الزمن 0** بالترتيب **P5 ... ,P2 ,P1** بطول زمني (CPU-burst) مقاس بالملي ثانية بأسبقية محددة وفقا للجدول الاتي (**غير قابلة للتوقف**).

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

باستخدام جدولة الاسبقية (كما هو موضح بالشكل التالي) فان :

متوسط زمن الانتظار = (1+6+16+18)/5 = 8.2ms

الاسبقية يمكن تحديدها :

- **داخليا عن طريق بعض الكميات القياسية** - محددات الوقت, متطلبات الذاكرة, عدد الملفات المفتوحة و النسبة بين متوسط زمن استخدم المعالج و زمن الادخال والاخراج.
- **والاسبقية الخارجية يتم وضعها عن طريق بعض المؤثرات التي تأتي من خارج نظام التشغيل** – اهمية العملية, نوع وقيمة المدفوع نظير استخدام الحاسوب وبعض المؤثرات السياسية.

يمكن ايضا ان تكون جدولة الاسبقية **قابلة للتوقف (Preemptive) وغير قابلة للتوقف (Nonpreemptive).**

- **قابلة للتوقف** اذا ظهرت عملية جديدة ذات **اسبقية اعلى** فإنها تعطى الاولوية.
- **غير قابلة للتوقف** اذا اعتمدت طريقة ان اي عملية جديدة يتم وضعها مباشرة على راس قائمة الجاهزية.

في **جدولة الاسبقية** فان **العمليات ذات الاسبقية الادنى قد لا يتم تنفيذها على الاطلاق** وخاصة اذا كان النظام مثقل بعدد العمليات التي تطلب التنفيذ وذات اسبقية عالية (**حالة Starvation**).

حل مشكلة الحرمان هو النضوج (Aging)، أي **كلما مر زمن طويل على عملية ذات أولوية دنيا نرفع أولويتها درجة**، وهكذا إذا مر وقت طويل على هذه العملية ستصبح ذات أولوية عليا ونكون بهذا قد مكناها من التنفيذ.

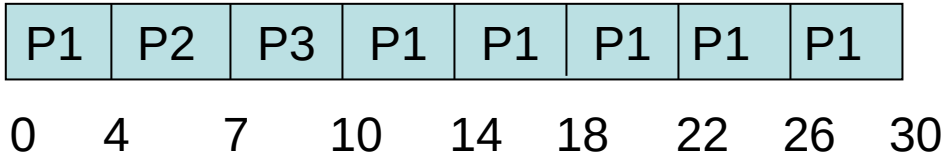
جدولة الراوند روبن (Round Robin RR)

صممت جدولة **RR** خصيصا لنظام **المشاركة الزمنية (Time-Sharing)**. وهي تشبه جدولة الرتل ولكنها تتميز بانها **قابلة للتوقف (Pre-emptive)**. حيث **تعطى كل عملية شريحة من الزمن (Quantum) للتنفيذ** وهي عادة ما تكون بين 10-100 ملي ثانية. حيث تعامل قائمة الجاهزية على اساس رتل دائري. حيث يخصص المعالج لكل عملية فترة من الزمن (1 شريحة من الزمن) فاذا لم تنهي العملية شغلها يتم توقيفها وتوضع في ذيل الرتل وينتقل المعالج لعملية اخرى. ام اذا انتهت قبل الزمن المخصص فإنها تسرح نفسها تلقائيا.

مثال 1: لدينا 3 عمليات وصلت بالترتيب **P1, P2, P3** بطول زمني (CPU-burst) مقاس بالملي ثانية **وبشريحة زمنية قدرها 4 ملي ثانية** وفقا للجدول الاتي.

Process	Burst time
P1	24
P2	3
P3	3

باستخدام جدولة الراوند روبن RR (كما هو موضح بالشكل التالي) فان :



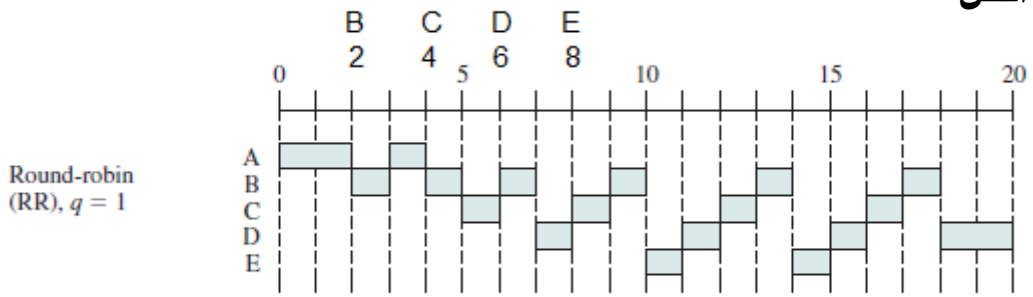
P2 سوف تنتظر 4 ملي ثانية, **P3** سوف تنتظر 7 ملي ثانية, **P1** سوف تنتظر (10- 4)= 6 وعلى ذلك فان

متوسط زمن الانتظار = (6+7+4)/3 = 17/3 = 5.66 ملي ثانية.

مثال 2 : اذا كان لدينا العمليات **A، B، C، D، E** قد وصلوا في زمن محدد بالجدول التالي وكل عملية تحتاج الى زمن تنفيذ ايضا محدد بالجدول.

- فأوجد **متوسط زمن الانتظار الاجمالي؟**
- اوجد (**Tr**) **Turnaround time**؟ وكذلك نسبة **Tr/Ts** حيث **Ts** يمثل **زمن التنفيذ.**
- أوجد **زمن انتهاء العمليات Finish time؟**

الحل



العمليات	زمن الوصول	زمن التنفيذ
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

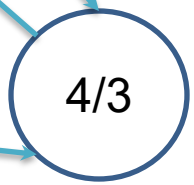
العمليات	زمن الانتظار
A	1
B	2+1+1+2+3+3=12-2=10
C	5+2+3+3=13-4=9
D	7+3+3+2=15-6=9
E	10+3=13-8=5
متوسط زمن الانتظار	34/5=6.8

حيث الرقم المطروح يمثل زمن الوصول

بجمع زمن التنفيذ من الجدول الاول مع زمن الانتظار في الجدول الثاني نحصل على الزمن المستغرق او الكلي(Turnaround time).كما هو موضح بالجدول التالي

RR $q=1$	A	B	C	D	E	
Finish Time	4	18	17	20	15	المتوسط
Turnaround Time (T_T)	4	16	13	14	7	10.80
T_T/T_S	1.33	2.67	3.25	2.80	3.50	2.71

العمليات	زمن الوصول	زمن التنفيذ
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



الزمن الكلي النسبي : هو نسبة الزمن الكلي لزمن المستغرق للتنفيذ وهو يوّضح زمن التأخيرٌ الذي حصل للعملية.

حيث، كلما كان زمن تنفيذ العملية طويل، كلما كان بإمكانها التسامح مع التأخير.

جدولة الارتال متعددة المستويات (Multilevel Queue Scheduling)

وهي **تستخدم في الحالات التي يمكن فيها فرز العمليات الي عدة مجموعات** فاذا كان التقسيم على اساس **العمليات التفاعلية** (Interactive system or Foreground) **وعمليات الدفعات** (Batch system Background) فان متطلبات النوعين ستكون مختلفة من حيث على سبيل المثال **زمن الاستجابة,** فالعمليات التفاعلية تتطلب زمن استجابة اسرع.

ولحل مثل هذا الخليط فان **خوارزمية الارتال متعددة المستويات تعتمد على تقسيم طابور الجاهزية الى عدة ارتال**, حيث يتم تخصيص العمليات تخصيصا دائم في احدى الارتال بناء على بعض **الخصائص- حجم الذاكرة المطلوبة, الاسبقية, نوع العملية.**

كل رتل له خوارزمية جدولة تحدد كيفية التعامل مع هذا الرتل, على سبيل المثال يمكن استخدام خوارزمية الراوند روبن (RR) في حالة العمليات التفاعلية و خوارزمية الرتل (FCFS) في حالة نظام الدفعات Batch system (background).

مثال : يبين المثال 5 ارتال مصنفة بأسبقية مرتبة حسب الترتيب الاتي:

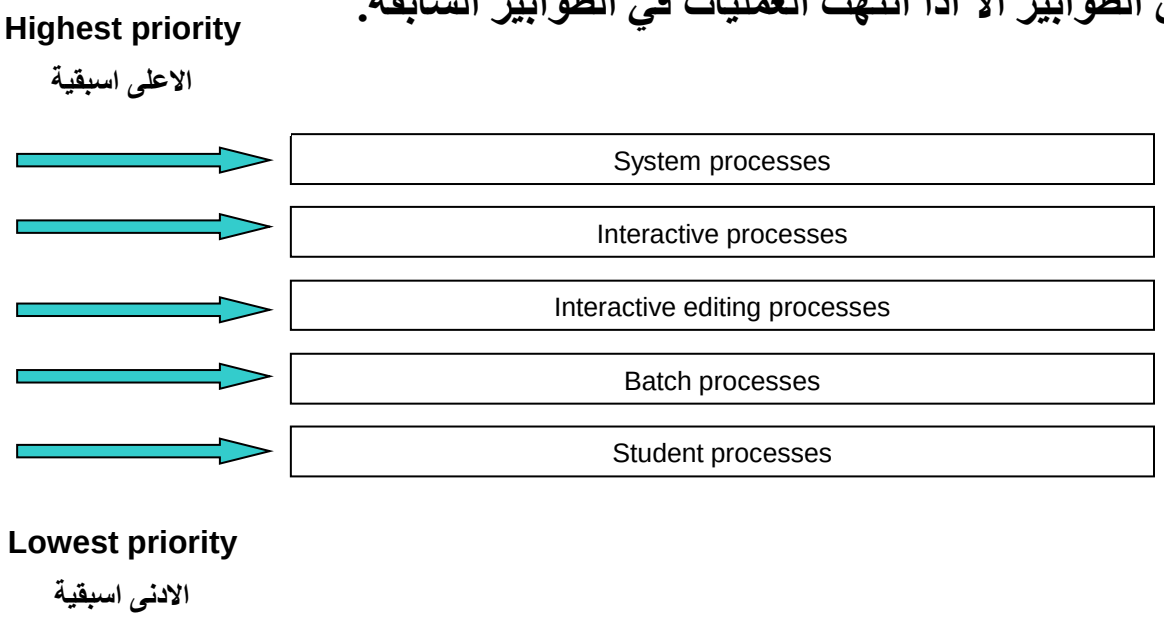
1. عمليات النظام (System Processes).
2. العمليات التفاعلية (Interactive Processes).
3. عمليات التحرير التفاعلية (Interactive Editing Processes).
4. عمليات الدفعات (Batch Processes).
5. عمليات الطلبة (Student Processes).

الشكل الاتي يبين جدولة الارتال متعددة المستويات.

في هذا النمط **فان الاسبقية تحتم انه لا يمكن تنفيذ عمليات الدفعات قبل ان تتم عمليات النظام، والعمليات التفاعلية، وعمليات التحرير التفاعلية قبل البدء في تنفيذ عمليات الدفعات (Batch process)**. واذا ما دخلت عملية جديدة ذات اسبقية اعلى من عملية الدفعات (وهي في حالة شغل) فانه يتم تخصيص المعالج للعملية ذات الاسبقية الاعلى على الفور (preemptive).

كما يمكن **تخصيص شريحه زمن كنسبة اجمالية من زمن المعالج لكل نوع** بحيث يتم توزيعها بين العمليات لكل رتل, على سبيل المثال **80%** من زمن المعالج يخصص الى العمليات التفاعلية (**RR**) و **20%** الى عمليات الدفعات (**FCFS**).

نلاحظ هنا عدم الانتقال بين الطوابير الا اذا انتهت العمليات في الطوابير السابقة.



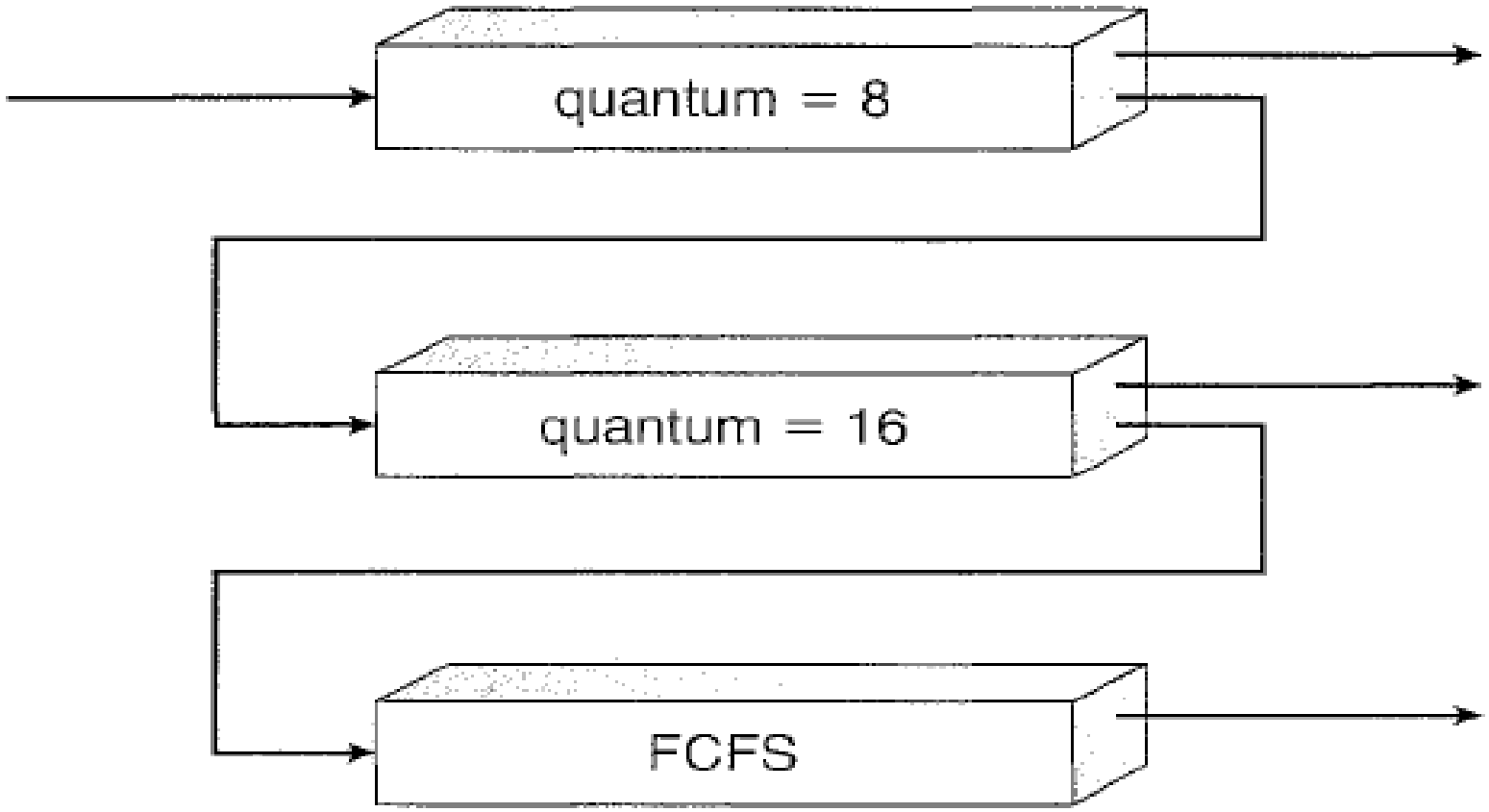
جدولة التغذية المرتدة في الارتال متعددة المستويات (Multilevel Feedback Queue Scheduling)

في **الجدولة السابقة وجود العمليات في الارتال عبارة عن وجود دائم** بمعنى انه يتم اختيار العملية دائما حسب الرتل الموجودة فيه **ولا يتم انتقالها الى اي رتل اخر**. وهذه ميزة تخفف من النشاط الاضافي (Overhead) ولكنها غير مرنة.

بينما **تسمح جدولة التغذية المرتدة في الارتال متعددة المستويات بانتقال العمليات بين الارتال**. وهي تعتمد على **تقسيم الارتال حسب زمن احتياج العمليات للمعالج (CPU-burst)** . فاذا كانت العملية تحتاج الى زمن كبير فإنها تنتقل الى المستوى الادنى التالي لها و توضع في نهاية القائمة. كذلك العمليات التي انتظرت طويلا يمكنها الانتقال الى المستوى الاعلى اسبقية, مما يؤدي الى اعطى العمليات ذات خاصية الادخال والاخراج (I/O-bound) الفرصة في استخدام المعالج. وبذلك **يمكن منع حالة الحرمان (Starvation).**

الشكل الاتي يفترض ان لدينا **3 ارتال 0-2** حيث يتم تنفيذ الارتال حسب الترتيب المعطى وذلك بإتمام العمليات الموجودة **بالرتل 0** اولا تليها العمليات الموجودة **بالرتل 1** وهكذا.

عند وصول اي عملية على سبيل المثال الى **الرتل 1** فانه يتم انتزاع (**preemptive**) المعالج من المستوى **الادنى (2)** لصالح هذه العملية (القادمة).


quantum = 8
quantum = 16
FCFS

حجم الحصة الزمنية Quantum Time

تحديد حجم الحصة او الشرحة الزمنية هو في الواقع شيء حرج بالنسبة لنظام التشغيل حتى يكون فعال. بمعنى هل حجم الحصة الزمنية يجب ان تكون طويلة او قصيرة، ثابتة ام متغيرة، متساوية لجميع المستخدمين او يحدد باستقلال لكل مستخدم.

كل هذه المعطيات تؤخذ بعين الاعتبار ليكون نظام التشغيل مناسبا لتوفير زمن استجابة مقبول. فاذا كان **حجم الحصة الزمنية كبيرة (طويلة)** فان كل عملية يتم تنفيذها الى النهاية، فان جدولة الراوند روبن (RR) تنتهي الى جدولة الطابور (FCFS).

وفي حالة ان يكون **حجم الحصة الزمنية يكون صغيرا جدا**، فان زمن سياق الانتقال من عملية الى اخرى يصبح متكررا بشكل يؤدي الى ابطاء النظام ككل وهو شغل اضافي (Overheads) مصاحب لعملية التبديل.

وبصفة عامة فانه يتم اختيار الحصة الزمنية بحيث تكون كبيرة للدرجة التي تسمح لأغلب الطلبات الغير مهمة بان تنتهي في حصة زمنية واحدة.

تمارين

1. أذكر ثلاث أمثلة من خوارزميات الجدولة ؟
2. إذا كان لدينا العمليات التالية مبين فيها ما تحتاجه من وقت داخل المعالج:

العملية	زمن الوصول	زمن العملية
P1	0	9
P2	2	6
P3	4	2
P4	5	4

إذا كان **زمن الانتظار = (12+2+0+5)/4= 4.75**، **اجب عن الاتي** :

- ما هي الخوارزمية المستخدمة
- أرسم شكل (**Gantt chart**) يوضح تنفيذ العمليات داخل المعالج.
- أثبت أن **متوسط زمن الانتظار = 4.75**

3. ما هو الحرمان (starvation)، وكيف نعالجه؟

4. بافتراض أن هنالك مجموعة من العمليات موضحة في الجدول التالي:

العملية	زمن الوصول	زمن العملية	الأولوية
P1	0	9	3
P2	2	7	1
P3	4	5	3
P4	6	10	4
P5	7	2	2

المطلوب

- رسم مخطط جانت (Gantt chart)
- حساب متوسط زمن الإنتظار
- حساب متوسط زمن الإكتمال للعمليات أعلاه باستخدام الخوارزميات التالية:
  - القادم أولا يخدم أولا (FCFS) بدون زمن وصول
  - القادم أولا يخدم أولا (FCFS) مع زمن الوصول
  - الأقصر أولا (SJF) الغير قابلة للتوقف (بدون زمن وصول)
  - الأقصر أولا (SJF) الغير قابلة للتوقف (مع زمن الوصول)
  - الأقصر أولا (SJF) القابلة للتوقف (مع زمن الوصول)
  - الأفضلية (الأهمية) الغير قابلة للتوقف
  - الأولوية (الأهمية) القابلة للتوقف مع زمن الوصول
  - التقسيم الزمني (RR) مع Q=2 بدون زمن وصول
  - التقسيم الزمني (RR) مع Q=2 مع زمن الوصول
  - ما هي أفضل خوارزمية من واقع النتائج التي تحصلت عليها ؟ ولماذا ؟